c Project Phase 4

Database Management 308N

Section 111

Team 1

Solomon Henry

Cayleigh Goberman

Jakeline Linares

Cheryl Kumah

Erick Perez

Marist College

School of Computer Science

Submitted to Dr. Reza Sadeghi

21 March 2024

Table of Contents

Table of Figures ……..……..... 3

Project Objective …...……..……..... 4

GitHub Repository Address 5

Review of Related Works …...……...……... 6

The Merits of Your Project 8

Entity Relationship Model (ER Model) 9

Enhanced Entity Relationship Model (EER Model) 11

Presentation Subjects

String Data Types 12

Data Type Default Values 13

Example Code 14

Database Development 17

References …….……...…... 23

Table of Figures

Entity Relationship Model (ER Model) 9

Enhanced Entity Relationship Model (EER Model) 11

Project Objectives

The project we will work on is Diary Management System (DMS Project Sample 1).

Project Objective: Create a Diary Management System that allows different users to record daily events and experiences. Including the ability to add and update records.

The system will contain

- Log In:
    - Admin user and password
    - Ability to change user and password
    - Admin should have ability to add user to DMS by creating a new username and password for normal users
    - Admin should have the ability to remove user by removing username, password and other corresponding data
- Diary Records:
    - Add/Remove/Edit/Search records
    - Include time, place, duration
        - Search based on those features and list all the results on the screen
- User friendly:
    - Task overlap warnings
    - Welcome page
    - Menu with all functions for user
    - Provide reports in tabular form
    - Exit function

GitHub Repository

GitHub Repository Link: https://github.com/Solomon-Henry/111_ProjectTitle_Team1.git

Review of Related Works

Sample 1: Schedule it – Diary Management Software [1]

"Diary management software is a digital tool that helps individuals and businesses organize appointments, tasks, and events in a centralized calendar system, streamlining scheduling and enhancing productivity."

Positives:

- This software simplifies various planning tools into a single platform, increasing efficiency and eliminates the need for multiple platforms.
- It's user-friendly and has easy to use features like drag and drop making data management quicker and simpler.
- The software is accessible on any device, which allows users to access it anytime and anywhere.
- It simplifies business tasks by automating report generation and timesheets.
- The software also offers unlimited diaries, which means it can be used in larger organizations.

Negatives:

- It may require a lot of training due to its advanced features, which can take time and additional resources.
- This software might be too complex for smaller groups/companies because of its many features.
- It relies on a stable internet connection, which may be a problem in areas with poor connectivity. It also means that the data cannot be accessed offline.
- The software does have a free version, however, the full package which includes all the features comes at a higher cost.
- Because it is cloud-based, there may be some concerns about its level of security and data privacy.

Sample 2: Adobe Express [2]

Adobe Express is an online journal-creation website that allows users to utilize various templates and resources in order to create their desired diary.

Positives:

- Adobe Express has a basic free version and a paid premium version if users wish to gain access to more advanced tools.
- There are plans curated for individuals, students, and businesses, which allows users to select which format best fits their needs.
- Numerous templates are available for users to employ based on how they want the diary to look.

- A section of the website is dedicated to tutorials in order to assist users in learning the available tools.
- The website has a section that focuses on social media scheduling.

Negatives:

- All diaries are stored on the Cloud; if a user wants to have the diary on the computer, they must download it.
- The diaries can only be edited on the website. Downloaded diaries are static and cannot be changed.
- The website contains numerous tools that assist in the creation of diary entries, but there are so many that it might take a user some time to master them.

Sample 3: Day One [3]

Day One is an application that can be used on all devices and allows users to create daily journal entries. It also lets users include any videos or pictures they want without having to worry about their privacy being compromised.

Positives:

- Free option for users that includes basic tools but can choose to purchase a subscription to have access to more advanced tools.
- Day One has security features to protect the user's data like passcodes, end-to-end encryption, and biometric security.
- Day one is accessible on almost every device with internet access.
- Users can have different journals for different purposes.
- Many customization options for the appearance of journal entries, including different fonts and formatting styles, allowing users to personalize their digital journal.

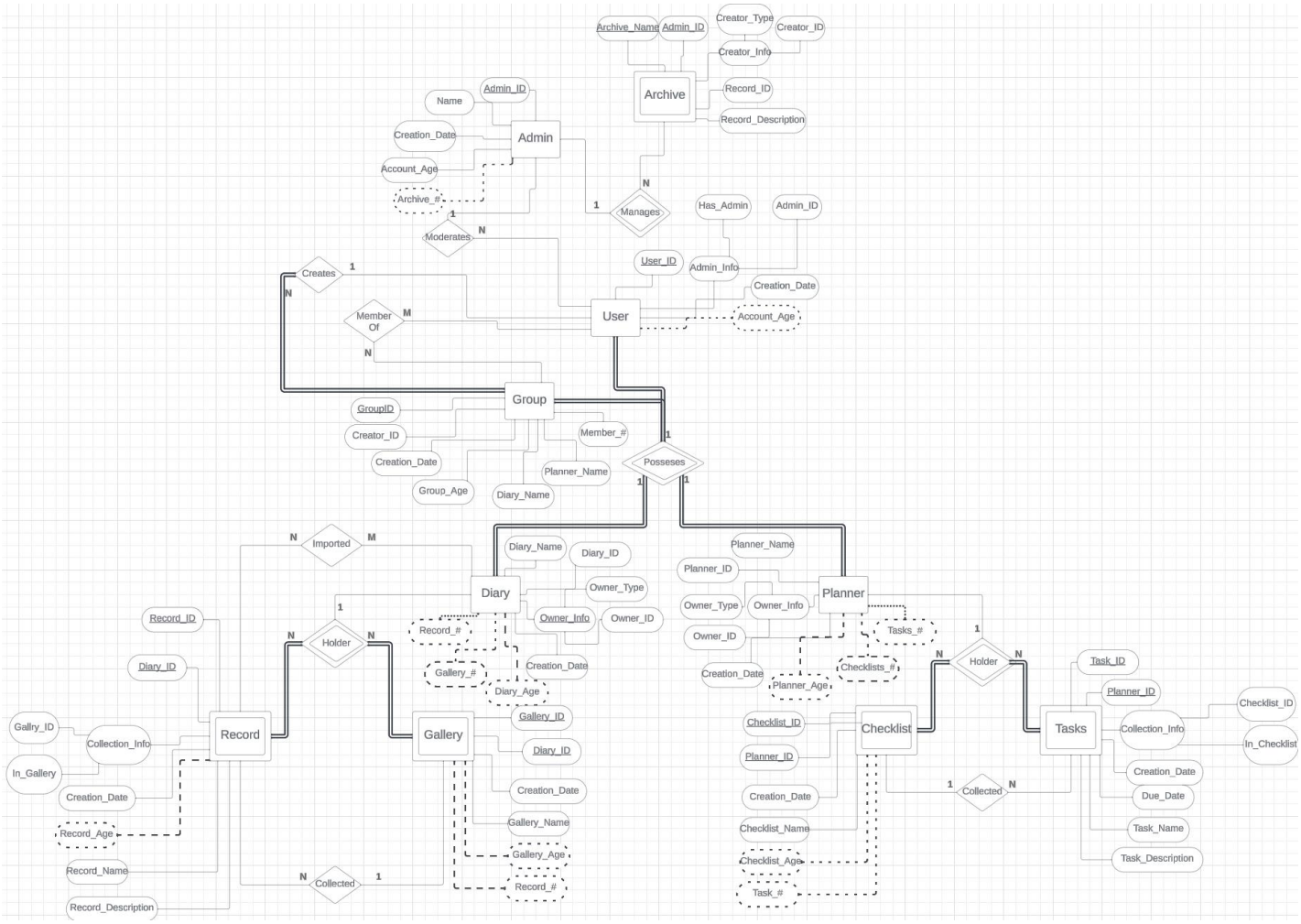
Negatives:

- The subscription can be a negative for people who want access to all the tools without having to pay or prefer one-time purchases.
- Day One depends on cloud services and internet access. Some users may not prefer this.
- Day One mainly focuses on personal journaling and doesn't really have much for collaborative use.

Merits of Our Project in Comparison:

These samples are all very similar to our Diary Management System (DMS) project, but also contain differences. First, our DMS is very alike to Sample 1, “Schedule It”. Similarly to the calendar system present in “Schedule It”, our DMS allows users to record daily events and save the time, place, and duration of the record [1]. However, our DMS is more for diary records; in comparison, this sample is more like a schedule, but both the DMS and the sample are intended to be user-friendly. For example, a menu displaying all functions for the user is available on both systems. Second, Sample 2, “Adobe Express”, is like our DMS because it’s an online journal-creation website intended to assist the user in the development of a desired diary. Our system also keeps diary records, but Adobe Express offers additional tutorials to provide users with the freedom to select a desired format and to assist users in learning all available tools [2]. Our DMS does not contain tutorials and has a default setup, but its user-friendly design includes task overlap warnings, a menu with functions for the user, and a tabular form setup, Sample 3, “Day One”, is an app that allows users to be able to create daily journals. Our DMS is also useful for daily journaling and will save all records with time, place and duration. Similarly, “Day One” has security features where users protect their individual accounts with usernames and passwords to enhance privacy [3]. Our DMS does not offer customization; instead, it provides a default setup in tabular form and includes username and password functionality. While these samples are similar to our project in some aspects, our DMS is proven to have numerous differences in the features offered. These differences ultimately allow this product to stand out in comparison to the competition, thus creating an important distinction in customers’ minds that will draw them to our product.

Entity Relationship (ER) Model



1. Describe how you created this mini world and how you selected the entities, attributes, relationships, participations, and cardinality.

First, you must identify the main entities in this mini world. This includes Admin, Archive, User, Group, Diary, Record, Gallery, Planner, Tasks, and Checklist. Each entity needs attributes to describe or characterize it. For example, attributes for the admin entity are AdminID, Name, CreationDate, etc. After the entities and their attributes were identified, the relationships within the world have to be identified. Relationships were established based on interactions between entities. For example, an Admin entity might manage multiple Archives and it establishes a relationship. Participations in relationships are also identified; an Admin might participate in the manage relationship with archive partially. One of the cardinalities identified was 1-to-many between the relationship Admin and Archive.

2. Provide a short description about each entity, attribute, relationship, participation, and cardinality.

Admin: Represents administrators within the system who manage the platform.

Archive: Stores archived data and managed by admins.

User: Represents customers who use the system and create diaries.

Group: Represents a collection of users who may collaborate on diaries.

Diary: Represents a virtual diary owned by a user or group that contains entries.

Record: Represents individual entries within a diary and contains information like names and descriptions.

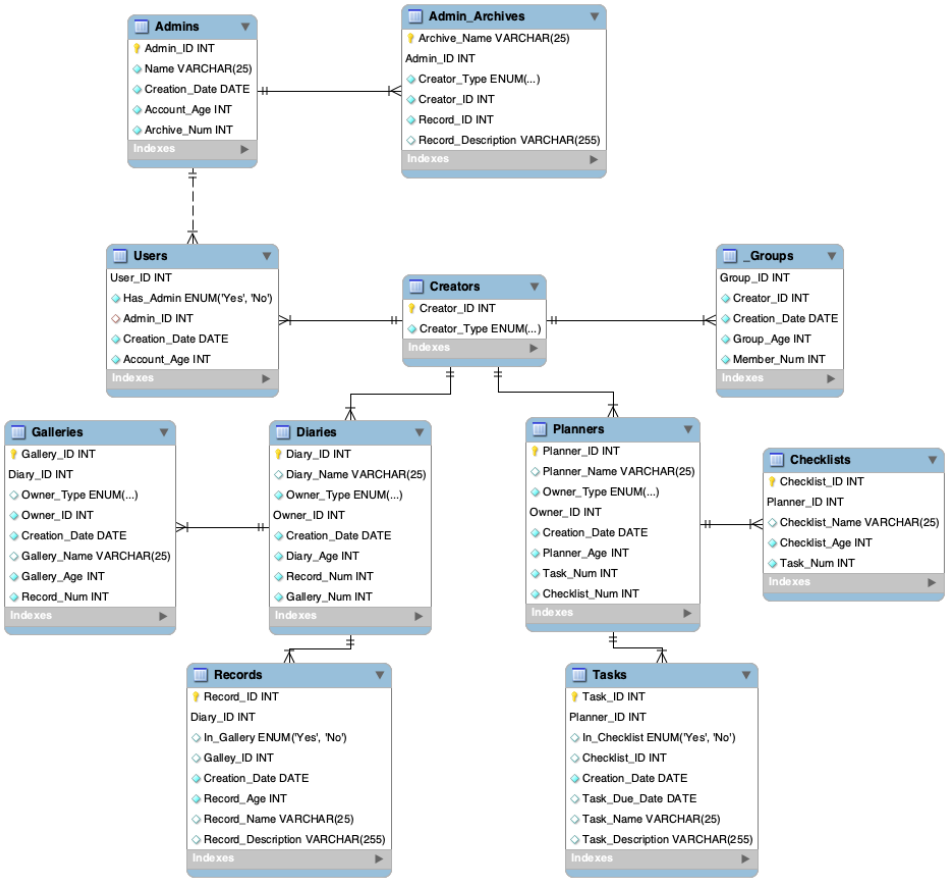
Gallery: Stores collection of records or entries for easy access or organization.

Planner: Represents a tool for organizing tasks or creating a checklist.

Tasks: Represents individual tasks or to-do items within a planner, with attributes like due dates or completion status.

Checklist: Represents a list of tasks within a planner for organizing activities.

Enhanced Entity Relationship (EER) Model



1. Provide a short description about keys and relationships.

In order to best provide security to our project, a variety of keys and relationships were utilized. As can be observed in the models, specific attributes and relationships were employed in areas best suited to their specific characteristics. For example, the “Diary” entity type is connected to attributes (Diary, Diary_ID, and Creation_Date), key attributes (Owner_Info), and weak key attributes (Record_#, Gallery_#, and Diary_Age). As such, numerous keys are employed to connect these attributes in order to create viable relationships. Additionally, these connections exhibit a variety of relationship types, such as one-to-one, many-to-many, and the most common type in our model, one-to-many.

2. Describe how you implemented these features on your EER model.

The EER model was created based on the ER model. As such, the various keys and relationships detailed in the latter were utilized to shape the former. The EER model is designed to represent the configuration of our product and assist us in the creation of that product. As such, the keys and relationships in the EER model will be crucial to the later phases of our project.

Presentation Subjects

Data Types specify the type of data that a column can contain [4].

String Data Types specify that the column can hold a string of data made up of characters [5].

String Data Types:

- CHAR: Creates a fixed length that one can declare after creating the table [6]
  - This length value can be between 0 and 255
  - If one skips declaring the length variable, it is automatically assigned to 1 [4]
  - If the user enters a string that is less than the declared value, the database will pad the remaining space to fill it to the correct value [6]
  - Ex: char(24) can hold up to 24 characters. If the user enters only 20 characters, the remaining 4 spaces will be padded.
- VARCHAR: Allows a varying length for the inputted string [6]
  - Length can be from 0 to 65,535
  - VARCHAR inputs are not padded: varchar(24) would only specify that the column can hold a **maximum** of 24 characters. 20 characters would be stored as 20 characters, not 24 (20 + 4 padding)
  - This can reduce the amount of storage required to save data
- BINARY: Stores a set string of bytes [7]
  - Similar to CHAR, except stores binary strings instead of nonbinary strings
- VARBINARY: Stores a varying string of bytes [7]
  - Similar to VARCHAR, except stores binary strings instead of nonbinary strings
  - Unlike CHAR and VARCHAR, the length of BINARY and VARBINARY is measured in byes, not characters
- BLOB: A binary large object that can hold a variable amount of data [8]
  - Four blob types: tinyblob, blob, mediumblob and longblob
    - They differ only in the maximum length of values they can hold
  - Blob values are treated as binary strings(byte strings), they have binary character set
- TEXT: Stores any kind of text data. Can contain both single-byte and multibyte characters [8]
  - Four test types: tinytext, text, mediumtext and longtext
  - Text values are treated as nonbinary strings(character strings), they have a character set other than binary
- ENUM: A string object with a value chosen from a list of permitted values that are enumerated explicitly in the column specification table at creation time [9]
  - The strings you specify as input values are automatically encoded as numbers.
  - The numbers are translated back to the corresponding strings in query results
  - Enum can have a maximum of 65,535 distinct elements
- SET: A string object that can have zero or more values each of which must be chosen from a list of permitted values specified when the table is created [10]
  - Set column values that consist of multiple set members must be separated by columns
  - A set column can have a maximum of 64 distinct members
  - For example, a column specified as SET('one', 'two') NOT NULL can have any of the values '"one' 'two' 'one,two'"

Importance of String Data Types [11]:

- String data types is the most common type to store text
- Strings are important when communicating information from the program to the user of the program

- A string can include digits and symbols as well
- The data type defines which operations can be performed to create transform and use the variable

Data Type Default Values:

What Are Default Values [12]?

- Predefined values that are assigned if no explicit value is provided when a table is created

Default Values Handling [12]

- Default Value is specified within the Default clause and can be a literal constant or expression
- Expressions need to be within parenthesis

Implicit Default Values Handling [12]

- A Data type specification with no explicit Default value is determined by MySQL
- If the column allows NULL values, MySQL specifies the column with an explicit Default NULL clause
- If the column does not allow NULL values, MySQL specifies the column with no explicit Default clause

Importance of Default Values [12]

- Reduces the burden on users to provide data for every field
- Provides a valid value for each field which prevents NULL or undefined values that can lead to errors
- Allows the user to use the default value temporarily until the explicit values are available

String Data Types Examples:

```
349
350 •  create database StringDataTypes;
351 •  use StringDataTypes;
352 •  drop table StringDataTypeEx;
353 •  create table StringDataTypeEx (
354    blobDescription BLOB,
355    textDescription TINYTEXT,
356    level ENUM('L_one', 'L_two', 'L_three'),
357    col SET('a', 'b', 'c')
358    );
359
360 •  insert into StringDataTypeEx(blobDescription, textDescription, level, col)
361     values ('Blob description', 'text','L_one', 'a');
362
363 •   select * from StringDataTypeEx;
```

100% 33:363 1 error found

Result Grid Filter Rows: Search Export:

blobDescription	textDescription	level	col
BLOB	text	L_one	a

StringDataTypeEx 2

```
/* CHAR and VARCHAR */
create table trees (
    tree_name char(20) key,
    scientific_name varchar(25),
    endangered char
);

insert into trees (tree_name, scientific_name, endangered) values
    ('White Oak', 'Quercus Alba', 'N'),
    ('Giant Sequoia', 'Sequoiadendron Giganteum','Y'),
    ('Red Maple', 'Acer Rubrum', 'N'),
    ('Frasier Fir','Abies Fraseri','Y');


  /* BINARY and VARBINARY */
  create table stars (
      list_number binary,
      star_name varchar(30),
      radius_x_that_of_sun varbinary(4),
      distance_from_sun_in_light_years varbinary(6),
      speed_in_km_per_sec binary(6)
  );

  insert into stars (list_number, star_name, radius_x_that_of_sun,
      distance_from_sun_in_light_years, speed_in_km_per_sec) values
      (1, 'Betelgeuse', 640, 500, 30),
      (2, 'Vega', 2.1, 25, 236),
      (3, 'Arcturus', 25, 36.7, 122),
      (4,'Spica', 7, 250, 199);
```

Data Type Default Values Examples:

```sql
CREATE DATABASE school;

USE school;

CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) DEFAULT 'Anonymous',
    age INT DEFAULT 18,
    enrollment_date DATE DEFAULT (CURRENT_DATE)
);



INSERT INTO students (name) VALUES ('Erick');
INSERT INTO students (age) VALUES (19);
INSERT INTO students (age, enrollment_date) VALUES (19, '2022-09-15');
select * from students;
```

```sql
-- Default Value Insertion
INSERT INTO students VALUES();
INSERT INTO students VALUES(DEFAULT,DEFAULT,DEFAULT,DEFAULT);
INSERT INTO students VALUES (DEFAULT(id), DEFAULT(name), Default(age), CURDATE());
```

```sql
ALTER TABLE students
MODIFY COLUMN age INT; -- DEFAULT NULL

-- Default value for age is now null
INSERT INTO students VALUES();
INSERT INTO students VALUES(DEFAULT,DEFAULT,DEFAULT,DEFAULT);
INSERT INTO students VALUES (DEFAULT(id), DEFAULT(name), Default(age), CURDATE());
```

```sql
ALTER TABLE students
MODIFY COLUMN age INT NOT NULL; -- NO DEFAULT CLAUSE

INSERT INTO students VALUES();
INSERT INTO students VALUES(DEFAULT,DEFAULT,DEFAULT,DEFAULT);
INSERT INTO students VALUES (DEFAULT(id), DEFAULT(name), Default(age), CURDATE()); -- Default function returns error
```

Database Development

Admins:

```
1   drop database Project_DMS;
2   create database if not exists Project_DMS;
3   use Project_DMS;
4
5   create table if not exists Admins (
6     Admin_ID int primary key,
7     Name varchar(25) not null,
8     Creation_Date date not null,
9     Account_Age int not null,
10    Archive_Num int not null
11  );
```

This admins table reflects the admins section in the ERR diagram. It contains five schemas: admin_id, name, creation_date, account_age, and archive_num. The primary key for this admins table admin_id which will help uniquely identify each admin in this Diary Management System. Not null refrains empty/null values.

Admin_Archives:

```
create table if not exists Admin_Archives (
  Archive_Name varchar(25) not null,
  Admin_ID int not null,
  Creator_Type enum('User', 'Group') not null default 'User',
  Creator_ID int not null,
  Record_ID int not null,
  Record_Description varchar(255),
  primary key (Archive_Name, Admin_ID),
  index (`Admin_ID` asc) visible,
  constraint
    foreign key (Admin_ID)
    references Admins (Admin_ID)
    on delete cascade);
```

This table addresses the Admin_Archives section outlined in our ERR diagram. Each schema is assigned a data type that most efficiently conveys the information contained within said schema. For example, Admin_II, Creator_ID, and Record_ID are assigned the int data type, while Archive_Name and record description are assigned the varchar data type. The Creator_Type schema is assigned the enum data type. Furthermore, Archive_Name was selected as the primary key of the table and Admin_ID was specified as a foreign key.

Users:

```
31
32    create table if not exists Users (
33      User_ID int not null,
34      Has_Admin enum('Yes', 'No') not null default 'No',
35      Admin_ID int,
36      Creation_Date date not null,
37      Account_Age int not null,
38      primary key (User_ID),
39      index (Admin_ID asc) visible,
40      constraint
41        foreign key (User_ID)
42        references Creators (Creator_ID)
43        on delete cascade,
44      constraint
45        foreign key (Admin_ID)
46        references Admins (Admin_ID)
47        on delete no action
48        on update no action);
49
```

This users table reflects the users section in the ERR diagram. It contains five schemas' user_id, has_admin, admin_id, creation_date, and account_age. The user_id is the primary key to uniquely identify the admin in this DMS. The admin_id is the foreign key that refers to the admins table. The on delete cascade specifies that when a row is deleted from the table all rows in the child table are also deleted. The index (Admin_ID asc) visible specifies that the admin_id is sorted in ascending order and retrieved really quickly and visible means the index is visible and not hidden.

_Groups:

```
18  create table _groups (
19      Group_ID int not null,
20      Group_Name varchar(50) not null,
21      Group_Description varchar(255) not null default 'User',
22      Owner_ID int not null,
23      Creation_Date date not null,
24      primary key (Group_ID),
25      foreign key (Owner_ID) references Planners(Planner_ID)
26  );
27
28  insert into _groups (Group_ID, Group_Name, Group_Description, Owner_ID, Creation_Date) values
29  (1, 'Test Group', 'A test group', 1, '2024-03-20');
```

In the _groups table, Group_ID is the primary key, Group_Name and Group_Description store the name and description, Owner_ID is a foreign key referencing the Planners table for the group owner, Creation_Date stores when the group was created and Group_Description has a default value of 'User'.

Creators:

```
CREATE TABLE IF NOT EXISTS Creators (
  Creator_ID INT NOT NULL,
  Creator_Type ENUM('User', 'Group') NOT NULL DEFAULT 'User',
  PRIMARY KEY (Creator_ID)
);
```

The Creators table was made to be similar to the creators section that was outlined in the ERR diagram to represent users and groups that create a diary. The second line identifies the attribute named Creator_ID and it's assigned the integer data types. The third line identifies the attribute named Creator_Type and it's assigned the enumeration data type, which restricts it to only two values which are User and Group. The primary key of this table is Creator_ID which will identify every unique creator.

Diaries:

```
CREATE TABLE IF NOT EXISTS Diaries (
  Diary_ID INT NOT NULL,
  Diary_Name VARCHAR(25) NULL DEFAULT NULL,
  Owner_Type ENUM('User', 'Group') NOT NULL DEFAULT 'User',
  Owner_ID INT NOT NULL,
  Creation_Date DATE NOT NULL,
  Diary_Age INT NOT NULL,
  Record_Num INT NOT NULL,
  Gallery_Num INT NOT NULL,
  PRIMARY KEY (Diary_ID, Owner_ID),
  INDEX (Owner_ID ASC) VISIBLE,
  CONSTRAINT
    FOREIGN KEY (Owner_ID)
    REFERENCES Creators (Creator_ID)
    ON DELETE CASCADE);
```

The Diaries table was made to be similar to the Diaries section that was outlined in the ERR diagram that represents diary entries. Diary_ID and Owner_ID are a composite primary key to identify unique diaries and they both take integer data types. Diary_Name attribute takes variable character data type with a max length of 25 letters. Owner_Type attribute takes the enumeration data type to specify whether it is a user or a group. Creation_Date attribute takes date data type to represent the date the diary made. Diary_Age attribute takes date data type to represent the age of the diary entry. Record_Num attribute takes the integer data type to represent the number of records. Gallery_Num attribute takes the integer data type to represent the number of galleries associated with the diary entry.

Records:

```
create table if not exists Records (
  Record_ID int not null,
  Diary_ID int not null,
  In_Gallery enum('Yes', 'No') default 'No',
  Galley_ID int,
  Creation_Date date not null,
  Record_Age int not null,
  Record_Name varchar(25),
  Record_Description varchar(255),
  primary key (Record_ID, Diary_ID),
  index (Diary_ID asc) visible,
  constraint
    foreign key (Diary_ID)
    references Diaries (Diary_ID)
    on delete cascade
);
```

The records table was designed to be similar to the Records section outlined in the ERR diagram. As with previous tables, appropriate data types were assigned to various schemas to best represent the information contained in them. The primary key of the table was assigned to the Record_ID schema while Diary_ID was assigned as a foreign key. While multiple records could potentially have the same name, they will each have a unique ID. As such, Record_ID was the best choice for the primary key assignment.

Galleries:

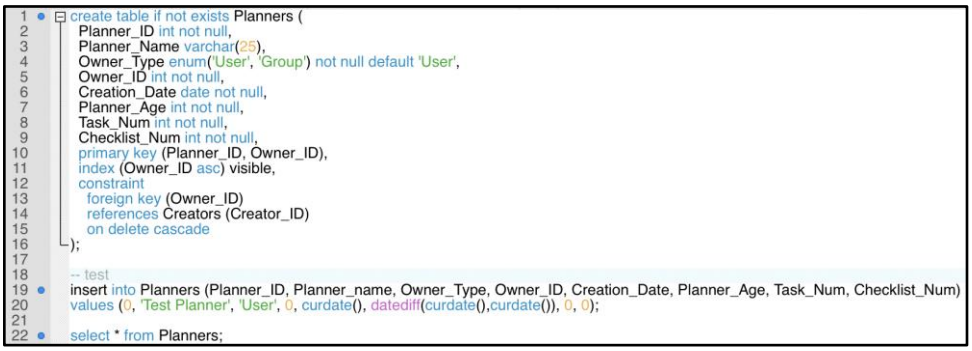
```
create table if not exists Galleries (
 Gallery_ID int not null,
 Diary_ID int not null,
 Owner_Type enum('User', 'Group') default 'User',
 Owner_ID int not null,
 Creation_Date date not null,
 Gallery_Name varchar(25),
 Gallery_Age int not null,
 Record_Num int not null,
 primary key (Gallery_ID, Diary_ID),
 index (Diary_ID asc) visible,
 constraint
  foreign key (Diary_ID)
  references Diaries (Diary_ID)
  on delete cascade
);

-- test
insert into Galleries (Gallery_ID, Diary_ID, Owner_Type, Owner_ID, Creation_Date, Gallery_Name, Gallery_Age, Record_Num)
values (0, 0, 'User', 0, curdate(), 'Test Gallery', datediff(curdate(), curdate()), 0);

select * from Galleries;
```

The Galleries table was made to reflect the Galleries section in the EER model, representing the collections of records that users may have. The primary key was made to be composed of both the Gallery_ID and Diary_ID attributes as Galleries are identified through their respective diary. Diary_ID has also been made to be the foreign key as a gallery can't exist without a corresponding diary. The Owner_Type attribute is an enumerated data type, with the two choices of User and Group, showcasing if the owner of the gallery is either a User or Group entity., with the following Owner_ID attribute specifying a specific one as an integer data type. Creation_Date attribute details the gallery's date of creation, with Gallery_Age detailing its age in days of last recording, and finally Record_Num being an integer data type showing the number of records stored within the gallery.

Planners:



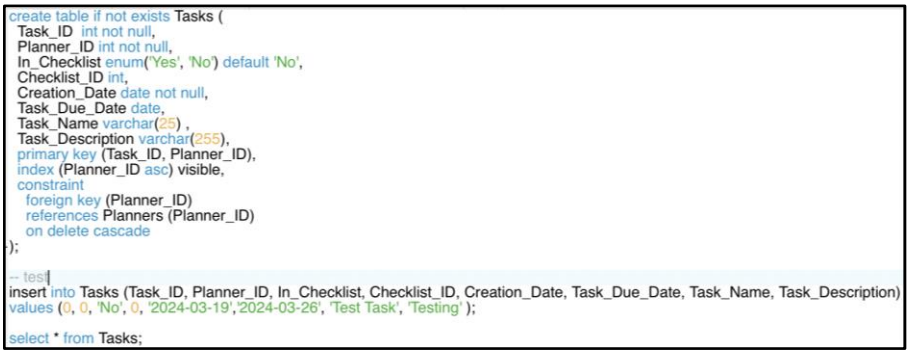
```
create table if not exists Planners (
    Planner_ID int not null,
    Planner_Name varchar(25),
    Owner_Type enum('User', 'Group') not null default 'User',
    Owner_ID int not null,
    Creation_Date date not null,
    Planner_Age int not null,
    Task_Num int not null,
    Checklist_Num int not null,
    primary key (Planner_ID, Owner_ID),
    index (Owner_ID asc) visible,
    constraint
      foreign key (Owner_ID)
      references Creators (Creator_ID)
      on delete cascade
);

-- test
insert into Planners (Planner_ID, Planner_name, Owner_Type, Owner_ID, Creation_Date, Planner_Age, Task_Num, Checklist_Num)
values (0, 'Test Planner', 'User', 0, curdate(), datediff(curdate(),curdate()), 0, 0);

select * from Planners;
```

The Planners table was made to reflect the Planners section in the EER model, representing the user's place to store important tasks. The primary key was made to be composed of both the Planner_ID and Owner_ID attributes as Plannersare identified through their respective owner. Owner_ID has also been made to be the foreign key as a gallery can't exist without a corresponding owner. The Owner_Type attribute is an enumerated data type, with the two choices of User and Group, showcasing if the owner of the planner is either a User or Group entity, with the following Owner_ID attribute specifying a specific one as an integer data type. Creation_Date attribute details the planner's date of creation, with Planner_Age detailing its age in days of last recording, and finally Task_Num and Checklist_Num both being integer data type attributes showing the number of tasks and checklist stored stored within the planner.

Tasks:



```
create table if not exists Tasks (
  Task_ID  int not null,
  Planner_ID int not null,
  In_Checklist enum('Yes', 'No') default 'No',
  Checklist_ID int,
  Creation_Date date not null,
  Task_Due_Date date,
  Task_Name varchar(25) ,
  Task_Description varchar(255),
  primary key (Task_ID, Planner_ID),
  index (Planner_ID asc) visible,
  constraint
    foreign key (Planner_ID)
    references Planners (Planner_ID)
    on delete cascade
);

-- test
insert into Tasks (Task_ID, Planner_ID, In_Checklist, Checklist_ID, Creation_Date, Task_Due_Date, Task_Name, Task_Description)
values (0, 0, 'No', 0, '2024-03-19','2024-03-26', 'Test Task', 'Testing' );

select * from Tasks;
```

The Tasks table was made to reflect the Tasks section in the EER model, representing the objectives that the user may wish to keep track of. The primary key was made to be composed of both the Task_ID and Planner_ID attributes as Tasks are identified through their respective planner. Planner_ID has also been made to be the foreign key as a task can't exist without a corresponding planner. The In_Checklist attribute is an enumerated data type, with the two choices of Yes and No, showcasing if the task is stored within a checklist or not, defaulting to no, with the following Checklist_ID attribute specifying a specific one as an integer data type if it is in a checklist. Creation_Date is a date type attribute storing when the task was created and Task_Due_Date is a date type attribute detailing a due date for the task if such a due date exists,

else it defaults to null. Task_Name and Task_Description each respectively store the tasks name and description as varchar data types, but are allowed to be null.

Checklists:

```
CREATE TABLE checklists (
    Checklist_ID int not null,
    Partner_ID int not null,
    Checklist_Done enum('Yes', 'No') default 'No',
    Creation_Date date not null,
    Task_Due_Date date,
    Task_Name varchar(50),
    Task_Description varchar(255),
    primary key (Checklist_ID, Partner_ID),
    foreign key (Partner_ID) references Planners(Planner_ID)
);

insert into Checklists (Checklist_ID, Partner_ID, Checklist_Done, Creation_Date, Task_Due_Date, Task_Name, Task_Description) values
(1, 1, 'No', '2024-03-19', '2024-03-26', 'Test Task', 'Testing');
```

As with previous tables, appropriate data types were assigned to various schemas to best represent the information contained in them. In this table Checklist_ID and Partner_ID form the composite primary key, Partner_ID is a foreign key referencing the Planners table, Checklist_Done is an enum field with 'Yes' or 'No' values, Creation_Date and Task_Due_Date are date fields, and Task_Name and Task_Description store details about the task.

References:

[1] https://www.scheduleit.com/diary-software.htm

[2] https://www.adobe.com/express/create/online-journal

[3] https://dayoneapp.com

[4] https://www.sqltutorial.org/sql-data-types/
[5] https://dev.mysql.com/doc/refman/8.0/en/string-type-syntax.html

[6] https://dev.mysql.com/doc/refman/8.0/en/char.html
[7] https://dev.mysql.com/doc/refman/8.0/en/binary-varbinary.html
[8] https://dev.mysql.com/doc/refman/8.0/en/blob.html
[9] https://dev.mysql.com/doc/refman/8.0/en/enum.html

[10] https://dev.mysql.com/doc/refman/8.0/en/set.html

[11] https://amplitude.com/blog/data-types

[12] https://dev.mysql.com/doc/refman/8.0/en/data-type-defaults.html